An Empirical Study of a Linear Regression Combiner on Multi-class Data Sets

Chun-Xia Zhang[1,2] and Robert P.W. Duin[2]

[1] School of Science and State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049, China
cxzhang@mail.xjtu.edu.cn

[2] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
r.duin@ieee.org

Abstract. The meta-learner *MLR* (Multi-response Linear Regression) has been proposed as a trainable combiner for fusing heterogeneous base-level classifiers. Although it has interesting properties, it never has been evaluated extensively up to now. This paper employs learning curves to investigate the relative performance of *MLR* for solving multi-class classification problems in comparison with other trainable combiners. Several strategies (namely, *Reusing*, *Validation* and *Stacking*) are considered for using the available data to train both the base-level classifiers and the combiner. Experimental results show that due to the limited complexity of *MLR*, it can outperform the other combiners for small sample sizes when the *Validation* or *Stacking* strategy is adopted. Therefore, *MLR* should be a preferential choice of trainable combiners when solving a multi-class task with small sample size.

Keywords: Ensemble classifier, Multi-response linear regression (*MLR*), Trainable combiner, Decision template (*DT*), Fisher linear discriminant (*FLD*).

1 Introduction

The task of constructing an ensemble classifier [1,2] can be broken into two steps, that is, generating a diverse set of base-level classifiers and combining their predictions. In general, there are two commonly used approaches to generate multiple base-level classifiers. One method is to train classifiers from different executions of the same learning algorithm through injecting randomness into the learning algorithm or manipulating the training examples, the input attributes and the outputs [3, 4, 5]. The obtained classifiers are often called *homogeneous* and they are typically combined by fixed combination rules such as weighted or unweighted voting. The other method is to apply some different learning algorithms to the same data set [6, 7, 8, 9, 10, 11, 12]. The derived classifiers are called *heterogeneous* and trainable combiners can generally merge them to result in a good ensemble classifier [7]. In the present study, we will focus on the heterogeneous base-level classifiers and trainable combiners.

J.A. Benediktsson, J. Kittler, and F. Roli (Eds.): MCS 2009, LNCS 5519, pp. 478–487, 2009.

With respect to trainable combiners, we are faced with the following problem: how to utilize the available data to train the models for both levels, the base-level classifiers as well as the combiner? Thus far, three strategies (that is, *Reusing*, *Validation* and *Stacking*) [13] have been proposed and they will be briefly described in Section 2.

In recent years, multi-response linear regression (*MLR*) has been recommended as a trainable combiner for merging heterogeneous base-level classifiers and there have been some variants of it [6,7,9,10,14]. Among the different methods related to *MLR*, the approach proposed in [7] may be the prominent one and some evidence has shown that it can handle multi-class problems very well [9]. To the best of our knowledge, however, the previous researchers only considered the situation that the training set size is supposed to be fixed and the *Stacking* method is employed to construct its meta-level data (namely, the data for training the combiner).

It is still unclear whether *MLR* will always keep its superiority with respect to different sample sizes and using different techniques to form its meta-level data. Thus, in this paper we employ learning curves to investigate the relative performance of *MLR* for solving multi-class classification problems in comparison with other combiners *FLD* (Fisher Linear Discriminant), *DT* (Decision Template) and *MEAN*. Several strategies are considered for using the available data to train the base-level classifiers and the combiner. The experimental results show that for small sample sizes, *MLR* can generally outperform the other combiners when the *Validation* or *Stacking* strategy is adopted. Meanwhile, the *Reusing* strategy should be avoided as much as possible anyway. When the sample size is large, however, there is little difference between the compared combiners no matter what strategy is employed to form the meta-level data.

The remainder of the paper is organized as follows. In Section 2, the working mechanism of *MLR* as well as some feasible strategies to utilize the given data to derive both base-level classifiers and combiner is introduced. Section 3 presents and discusses the experimental studies conducted on some multi-class data sets. Finally, the main conclusions are given in Section 4.

2 *MLR* and Strategies to Use the Training Data

2.1 Working Mechanism of *MLR*

Consider a given set $\mathscr{L} = \{(y_n, \mathbf{x}_n)\}_{n=1}^{N}$, where y_n is a class label taking value from $\Phi = \{\omega_1, \omega_2, \cdots, \omega_m\}$ and $\mathbf{x}_n \in R^d$ is a vector representing the attribute values of the nth example. Suppose that a set $\mathscr{C} = \{C_1, C_2, \cdots, C_L\}$ of L base-level classifiers is generated by applying the heterogeneous learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ to $\mathscr{L}$ and the prediction of $C_i (i = 1, 2, \cdots, L)$ when applied to an example $\mathbf{x}$ is a probability distribution vector

$$\begin{aligned} \mathbf{P}_{C_i}(\mathbf{x}) &= (P_{C_i}(\omega_1|\mathbf{x}), P_{C_i}(\omega_2|\mathbf{x}), \cdots, P_{C_i}(\omega_m|\mathbf{x}))^T \\ &\triangleq (P_1^i(\mathbf{x}), P_2^i(\mathbf{x}), \cdots, P_m^i(\mathbf{x}))^T, \ i = 1, 2, \cdots, L, \end{aligned} \tag{1}$$

where $P_j^i(\mathbf{x})$ denotes the probability that the example $\mathbf{x}$ belongs to class ω_j as estimated by the classifier C_i. Furthermore, we define $\mathbf{P}(\mathbf{x})$ as an mL-dimensional column vector, namely,

$$\begin{aligned}\mathbf{P}(\mathbf{x}) &= (\mathbf{P}_1^T(\mathbf{x}), \mathbf{P}_2^T(\mathbf{x}), \cdots, \mathbf{P}_L^T(\mathbf{x}))^T \\ &= (\underbrace{P_1^1(\mathbf{x}), \cdots, P_m^1(\mathbf{x})}_{\text{Classifier } C_1}, \underbrace{P_1^2(\mathbf{x}), \cdots, P_m^2(\mathbf{x})}_{\text{Classifier } C_2}, \cdots, \underbrace{P_1^L(\mathbf{x}), \cdots, P_m^L(\mathbf{x})}_{\text{Classifier } C_L})^T \end{aligned} \quad (2)$$

Based on the intermediate feature space constituted by the outputs of each base-level classifier, the *MLR* method [7] firstly transforms the original classification task with m classes into m regression problems: the problem for class ω_j has examples with responses equal to one when they indeed have class label ω_j and zero otherwise. For each class ω_j, *MLR* selects only $P_j^1(\mathbf{x}), P_j^2(\mathbf{x}), \cdots, P_j^L(\mathbf{x})$, the probabilities that $\mathbf{x}$ belongs to ω_j predicted by the base-level classifiers $C_1, C_2, \cdots, C_L$, as the input attributes to establish a linear equation

$$LR_j(\mathbf{x}) = \sum_{i=1}^{L} \alpha_j^i P_j^i(\mathbf{x}),\ j = 1, 2, \cdots, m, \quad (3)$$

where the coefficients $\{\alpha_j^i\}_{i=1}^L$ are constraint to be non-negative and the non-negative-coefficient least-squares algorithm described in [15] is employed to estimate them. When classifying a new example $\mathbf{x}$, we only need to compute $LR_j(\mathbf{x})$ for all the m classes and assign it to the class ω_k which has the greatest value.

2.2 Strategies to Use the Training Data

In order to estimate the coefficients $\{\alpha_j^i\}_{i=1}^L$ in formula (3), or more generally, to train the combiner *MLR*, we have to form the meta-level data. In practical applications, however, we are only given a single set $\mathscr{L}$ which should be used to train the base-level classifiers as well as the combiner. In this situation, there may be three feasible approaches(*Reusing*, *Validation* and *Stacking*) to make full use of $\mathscr{L}$ to construct an ensemble classifier.

The *Reusing* strategy simply applies the given learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ to $\mathscr{L}$ to train the base-level classifiers $C_1, C_2, \cdots, C_L$ which are then used to predict the examples in $\mathscr{L}$ to form the meta-level data. Since the same set $\mathscr{L}$ is used to derive both base-level classifiers and combiner, the obtained combiner will inevitably be biased.

The *Validation* strategy splits the training set $\mathscr{L}$ into two disjoint subsets, one of which is used to derive the base-level classifiers $C_1, C_2, \cdots, C_L$ and the other one is employed to construct the meta-level data.

The *Stacking* strategy utilizes the cross-validation method to form the meta-level data. Firstly, the training set $\mathscr{L}$ is partitioned into K disjoint subsets $\mathscr{L}_1, \mathscr{L}_2, \cdots, \mathscr{L}_K$ of almost equal size. In order to obtain the base-level predictions on examples in $\mathscr{L}_k$, say, $\mathscr{L}_k' = \{(y_i, \mathbf{P}^T(\mathbf{x}_i)) | (y_i, \mathbf{x}_i) \in \mathscr{L}_k\}$, the learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ are applied to the set $\mathscr{L} \backslash \mathscr{L}_k$ to derive the classifiers $\{C_{j,k}\}_{j=1}^L$ which are then used to predict the examples in $\mathscr{L}_k$. After repeating

this process K times (once for each set $\mathscr{L}_k$), we can obtain the final meta-level data set $\mathscr{L}^{CV} = \bigcup_{k=1}^{K} \mathscr{L}'_k$. The readers can refer to [2, 6, 7, 13] for more details about this technique.

It is worthwhile to mention that for a new example $\mathbf{x}$, there may be two different ways to construct the input $\mathbf{P}(\mathbf{x})$ of the combiner trained by the *Stacking* method. The *Stacking I method*, commonly utilized by many researchers [16, 6, 7, 9, 10], is to retrain the base-level classifiers $C_1, C_2, \cdots, C_L$ on the full training set $\mathscr{L}$ to produce $\mathbf{P}(\mathbf{x})$. The *Stacking II method*, proposed in [13], applies all the classifiers $\{C_{j,k}\}_{j=1}^{L}{}_{k=1}^{K}$ which are trained in the process of cross-validation to predict $\mathbf{x}$. For each type of base classifier, it averages the predictions that are obtained in each fold, namely, $\bar{C}_j(\mathbf{x}) = (1/K) \sum_{k=1}^{K} C_{j,k}(\mathbf{x})$ and forms the input of the combiner as $\mathbf{P}(\mathbf{x}) = (\bar{C}_1^T(\mathbf{x}), \bar{C}_2^T(\mathbf{x}), \cdots, \bar{C}_L^T(\mathbf{x}))^T$.

3 Experimental Studies

Because learning curves can give a good picture to study the performance of an algorithm at various sample sizes, in this section we employ them to investigate the relative performance of *MLR* for solving multi-class classification tasks in comparison with several other combiners. Furthermore, each of the different strategies described in subsection 2.2 will be considered here to employ the given set to train the models for both levels.

3.1 Experimental Setup

We conducted experiments on a collection of 10 multi-class data sets from the UCI repository [17]. The data sets and some of their characteristics are summarized in Table 1.

Table 1. Summary of the data sets used in the experiments

Data set	# Examples	# Attributes	# Classes	Training size (Per class)	Test size (Per class)
Abalone	4177	10	3	5,10,20,30,50,80,100	500
Cbands	12000	30	24	10,20,30,50,80,100	100
Digits	2000	240	10	5,10,20,30,50,80,100	50
Letter	20000	16	26	10,20,30,50,80,100	200
Pendigits	7494/3498	16	10	5,10,20,30,50,80,100	3498(total)
Satellite	6435	36	6	5,10,20,30,50,80,100	500
Segmentation	2310	19	7	5,10,20,30,50,80,100	100
Vehicle	846	18	4	5,10,20,30,50,80,100	50
Vowelc	990	12	11	10,20,30,50	30
Waveform	5000	21	3	5,10,20,30,50,80,100	500

We totally considered 7 different types of base-level classifiers: Fisher linear discriminant (*fisherc*), Parzen density classifier (*parzenc*), Nearest neighbor (*knnc*), Logistic linear classifier (*loglc*), Nearest mean (*nmc*), Decision tree (*treec*), Support vector classifier (*svc*) and 4 different combiners: Multi-response linear regression (*MLR*), Fisher linear discriminant (*FLD*), Decision template

(DT) and Mean ($MEAN$), which were all implemented with PRTools[1]. The outputs of each base-level classifier were scaled to fall into a [0,1] interval so that the intermediate feature space is constituted in a homogeneous way. Considering the fact that we were combining the base-level classifiers with posterior probabilities or confidences as their outputs, the voting based combiners were not included. Furthermore, the parameters included into the base-level classifiers and the combiners were all taken to be their default values in PRTools. With the given base-level classifiers, two batches of experiments (one uses the first four ones and the other considers all the seven ones) were conducted to study the influence of the number of base-level classifiers on the relative performance of the combiners. The different strategies described in subsection 2.2 were respectively considered to utilize the available data to train base-level classifiers and combiners, in which the *Validation* method uses a 50%/50% split and the two *Stacking* methods employ 10-fold cross-validation to form the meta-level data.

We estimated the learning curves in the following way to understand the behavior of different combiners at various sample sizes. For each data set (except for "Pendigits") listed in Table 1, a training set and an independent test set with desired sizes were randomly sampled. As for the "Pendigits" data set which has separate training and testing data, all of its testing data was used as the test set. On each of the obtained training set, the base-level classifiers and the combiner were constructed according to each of the strategies described in subsection 2.2. The trained combiner was then executed on the test set and the estimated classification error was utilized to evaluate its performance. It should be noted that all steps required for building the base-level classifiers and the combiner, including the cross-validation utilized by the *Stacking* method, were performed on the training set only. For each training set size, the above process was repeated 10 times and the obtained results were averaged.

3.2 Results and Discussion

We firstly considered the cases that an ensemble classifiers is composed of 4 base-level classifiers. For each combiner, the mean of the test errors over 10 replications (standard deviation also shown) was plotted as a function of the sample size. Due to the limited space of this paper, only some representative plots obtained on "Cbands", "Digits" and "Satellite" data sets are presented here but the other ones are available from the authors. From these plots, the following observations can be made:

- For small sample sizes, *MLR* can generally perform better than the other combiners when the *Validation* or *Stacking* method is adopted. However, the superiority of *MLR* is not very obvious when the classification task has many classes ("Cbands") or high dimensionality of the input space ("Digits"), which may be caused by the inadequate diversity among the linear models established by *MLR*.

[1] PRTools is a Matlab Toolbox for Pattern Recognition and it can be freely downloaded from the PRTools website, http://www.prtools.org.

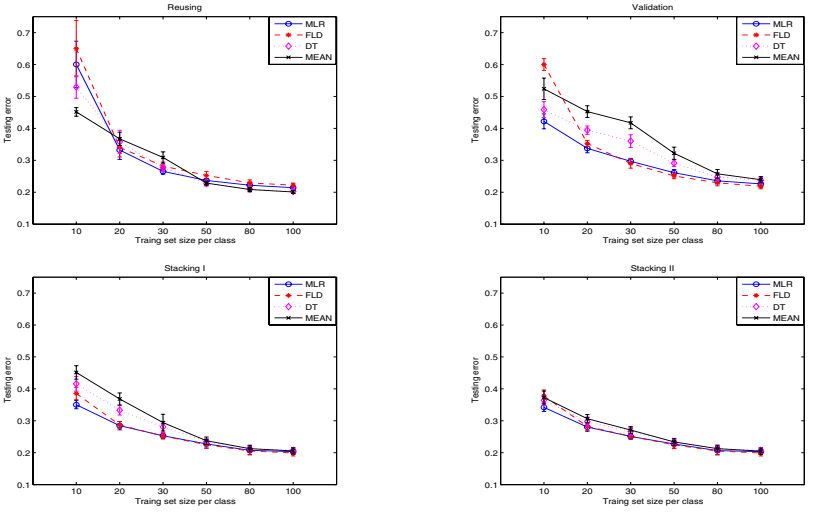

Fig. 1. On "**Cbands**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner

- When the sample size is large, there is little difference between the compared combiners no matter what strategy is used to form the meta-level data.
- The *Reusing* strategy should be avoided as much as possible anyway except for large sample sizes because the test errors corresponding to it are generally larger than those obtained by using the *Validation* or *Stacking* technique.
- Some combiners are observed in Fig. 2 to exhibit a dramatic error in some situations, which can be explained as the bad performance of base-level classifier *fisherc* or combiner *FLD* when the training set size is comparable to the dimension of the feature space [13, 18].
- For each combiner, the results obtained by *Stacking II* are slightly better than those derived by *Stacking I*. The reasons for this may be due to the fact that *Stacking II* benefits from more robust base-level classifiers [13].

In order to see clearly the relative performance of the compared combiners on the used 10 data sets, Table 2 provides some comparative summaries of the mean test errors for the combiner *MLR* in comparison with other ones. Here, the geometric means of error ratios and significant Wins-Losses of *MLR* compared with other combiners at each considered sample size were listed and these statistics were computed in the following way. Take the notation "*MLR/FLD*" as an example, at each sample size we firstly computed the error ratio of *MLR* to *FLD* on each data set, and then calculated the geometric mean of the obtained error ratios across all the data sets. Therefore, the value smaller than 1 indicates the better performance of *MLR*. With respect to the Win-Loss statistic, a paired t-test

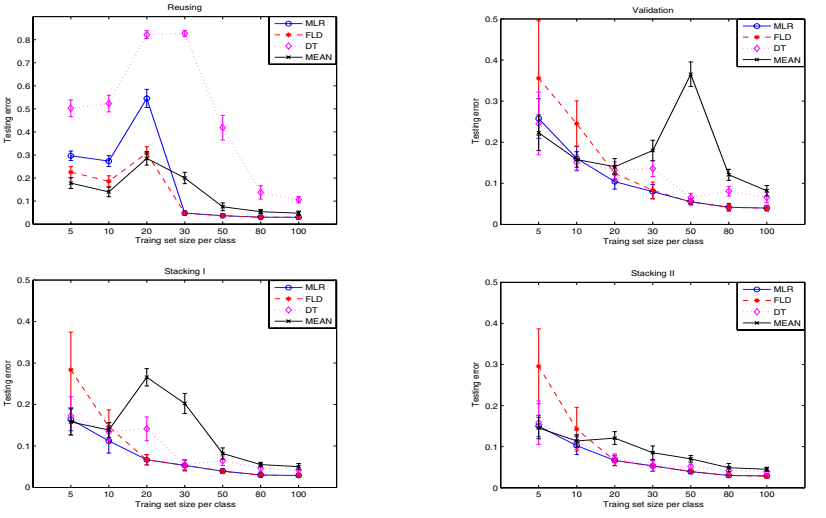

Fig. 2. On "**Digits**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner

was utilized to check whether the performance of *MLR* is significantly better than that of the other ones at the significance level 0.05 for each combination of data set and sample size and the numbers listed in the table should be read as the number of data sets on which *MLR* performs significantly better than *FLD*. From the results reported in Table 2, we can draw almost the same conclusions as those obtained from the previous figures.

Table 2. On the used 10 data sets, geometric means of error ratios as well as significant Wins-Losses of *MLR* in comparison with other combiners (4 base-level classifiers)

		Training set size per class						
		5	10	20	30	50	80	100
Reusing method	*MLR/FLD*	0.9842 3-2	0.9898 5-1	1.0269 4-1	0.9757 4-0	0.9760 2-0	0.9952 4-1	0.9886 3-1
	MLR/DT	1.0116 1-3	1.0375 1-6	1.0532 2-4	0.7557 3-5	0.7810 2-4	0.8404 3-5	0.8840 3-5
	MLR/MEAN	1.2563 0-6	1.2221 0-8	1.1510 1-5	0.8289 4-2	0.9123 2-4	0.9066 3-5	0.9380 4-5
Validation method	*MLR/FLD*	0.7045 6-0	0.7201 9-0	0.9517 5-2	1.0305 1-2	1.0367 1-5	1.0318 1-4	1.0371 0-3
	MLR/DT	0.8454 4-0	0.8718 9-0	0.8897 5-0	0.8717 5-2	0.9150 5-1	0.8721 5-0	0.9004 4-1
	MLR/MEAN	0.9051 3-0	0.8391 7-1	0.7779 7-0	0.7496 8-0	0.7070 7-0	0.7909 5-1	0.8326 5-1
Stacking I method	*MLR/FLD*	0.6983 7-0	0.8535 10-0	0.9782 2-1	0.9908 1-1	1.0158 1-4	1.0211 0-4	1.0316 0-4
	MLR/DT	0.8290 5-0	0.8569 6-0	0.8495 5-0	0.9239 4-1	0.9162 4-0	0.9037 4-1	0.9123 4-1
	MLR/MEAN	0.8922 3-0	0.8091 6-0	0.7500 5-0	0.7622 5-1	0.8525 6-1	0.8532 4-1	0.8620 4-1
Stacking II method	*MLR/FLD*	0.6919 7-0	0.8709 10-0	0.9869 2-1	0.9963 1-1	1.0171 0-3	1.0253 0-3	1.0347 0-5
	MLR/DT	0.8499 4-0	0.9041 5-0	0.9377 4-0	0.9579 4-1	0.9385 6-1	0.9213 5-0	0.9216 4-0
	MLR/MEAN	0.9299 3-2	0.8684 4-1	0.8513 4-1	0.8728 4-1	0.8743 6-1	0.8821 4-1	0.8816 4-1

Table 3. On the considered 8 data sets, geometric means of error ratios as well as significant Wins-Losses of *MLR* in comparison with other combiners (7 base-level classifiers)

		Training set size per class						
		5	10	20	30	50	80	100
Reusing method	*MLR/FLD*	1.0750 2-3	0.9784 4-1	0.9973 2-1	0.9503 4-1	0.9619 3-0	0.9722 1-0	0.9676 3-0
	MLR/DT	1.2238 0-5	1.1786 1-6	1.1888 1-5	0.8278 1-4	0.8078 3-3	0.8219 3-3	0.8534 4-3
	MLR/MEAN	1.4937 0-7	1.3714 1-7	1.4900 1-5	1.0298 1-3	0.9185 3-3	0.8889 4-3	0.8825 3-3
Validation method	*MLR/FLD*	0.8298 4-0	0.5536 8-0	0.7442 7-0	0.8878 6-1	1.0258 1-1	1.0177 0-1	1.0102 0-1
	MLR/DT	0.9410 1-0	0.9812 4-1	0.9786 3-1	0.9476 3-0	0.9140 4-0	0.8784 4-0	0.8663 5-0
	MLR/MEAN	1.0349 1-2	0.9104 6-0	0.8831 4-0	0.8355 8-0	0.7993 6-0	0.8176 5-0	0.8145 6-0
Stacking I method	*MLR/FLD*	0.5368 7-0	0.7356 8-0	0.9036 7-0	0.9452 3-0	1.0042 1-1	1.0343 0-2	1.0134 0-2
	MLR/DT	0.9759 2-1	0.9915 2-1	0.9259 4-0	0.9025 3-0	0.8712 5-1	0.8441 5-1	0.8240 5-0
	MLR/MEAN	1.0112 0-1	0.9481 4-0	0.8764 5-0	0.8508 7-0	0.8516 5-0	0.8110 5-0	0.7911 6-0
Stacking II method	*MLR/FLD*	0.5146 7-0	0.7427 8-0	0.9111 4-0	0.9650 2-1	1.0365 0-3	1.0266 0-2	1.0382 0-2
	MLR/DT	0.9498 0-0	0.9866 1-0	0.9547 4-0	0.9085 4-0	0.8755 6-0	0.8335 4-0	0.8267 5-0
	MLR/MEAN	1.0231 0-0	0.9440 4-0	0.8795 4-0	0.8289 5-0	0.7653 7-0	0.8091 5-0	0.8042 5-0

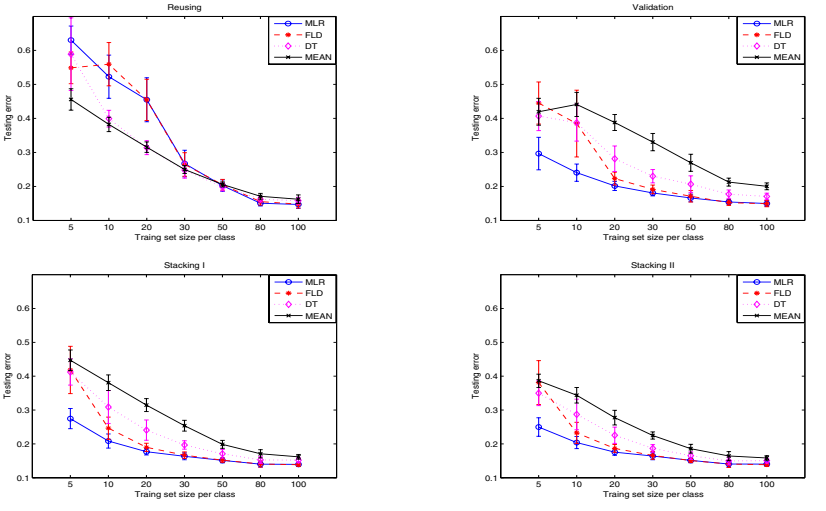

Fig. 3. On "**Satellite**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner

As for the experiments conducted with the 7 base-level classifiers, we reported the obtained results in Table 3 whose format is identical to that of Table 2. Considering that it takes too much time to conduct experiments on "Cbands" and "Letter" data sets, only the remaining 8 ones were taken into account in this batch of experiments. Through comparing the results listed in Tables 2 and 3, it seems that the superiority of *MLR* over the other combiners is more significant when more base-classifiers are used to form an ensemble classifier since *MLR* is observed to loss less times in this case.

3.3 Complexity Analysis

Now let us analyze the complexity of each considered combiner in terms of the number of parameters to be estimated.

Apparently, the combiner *MEAN* have not any parameters to estimate since it simply averages the probability distributions predicted by each base-level classifier and assigns $\mathbf{x}$ to the class having the largest probability. Because the combiner *MLR* needs to establish m linear models and each of them has L parameters, there are totally mL parameters required to be estimated.

If we use $\overline{\mathbf{P}}^k = (P^1_{k,1}, P^1_{k,2}, \cdots, P^1_{k,m}, P^2_{k,1}, P^2_{k,2}, \cdots, P^2_{k,m}, \cdots, P^L_{k,1}, P^L_{k,2}, \cdots, P^L_{k,m})^T$ to denote the mean vector of class ω_k in the intermediate feature space, the combiner DT estimates the class label of $\mathbf{x}$ as

$$\omega_{dt}(\mathbf{x}) = \underset{1\le k\le m}{\operatorname{argmin}} \sum_{i=1}^{L}\sum_{j=1}^{m}[P^i_{k,j} - P^i_j(\mathbf{x})]^2 = \underset{1\le k\le m}{\operatorname{argmin}} ||\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x})||^2. \tag{4}$$

Thus, there are m^2L parameters to be estimated.

As for the combiner *FLD*, it decides the class label of $\mathbf{x}$ as

$$\omega_{fld}(\mathbf{x}) = \underset{1\le k\le m}{\operatorname{argmin}} (\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x}))^T \Sigma^{-1} (\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x})), \tag{5}$$

here Σ indicates the sample estimate of the $mL \times mL$ covariance matrix supposed to be common for each class. Since the combiner *FLD* utilizes one-against-all strategy to solve a multi-class task, it needs to estimate $m(\frac{mL(mL+1)}{2} + 2mL)$ parameters in total.

Based on the above analysis, we can see that the compared combiners can be ordered from simple to complex as *MEAN*, *MLR*, *DT* and *FLD*. Thus, one of the reasons for the better performance of the combiner *MLR* than that of *DT* and *FLD* at small sample sizes may be attributed to its limited complexity. As for the advantage of *MLR* over *MEAN*, it may be due to the fact that *MLR* takes into account more information in the data whereas *MEAN* does not.

4 Conclusions

In this paper, we utilized learning curves to investigate the relative performance of *MLR* for solving multi-class problems in comparison of other trainable combiners *FLD*, *DT* and the fixed combiner *MEAN*. The *Reusing*, *Validation* and two versions of *Stacking* method were respectively considered for using the given data to train the base-level classifiers as well as the combiner. The experimental results show that *MLR* can outperform the other combiners for small sample sizes when *Validation* or *Stacking* method is employed. Meanwhile, the *Reusing* strategy should be avoided as much as possible anyway. When the sample size is large, however, there is little difference between the compared combiners no matter what strategy is employed to form the meta-level data. As for the two

Stacking methods, *Stacking II* may be preferred over *Stacking I* for its robustness and relatively smaller computational cost.

Acknowledgements. This research was supported in part by China State Scholarship Fund, National Basic Research Program of China (973 Program) (Grant No. 2007CB311002) and National Natural Science Foundations of China (Grant Nos. 60675013 and 10531030). This work was conducted when the first author was in Pattern Recognition Group, Delft University of Technology.

References

1. Kuncheva, L.I., Bezdek, J.C., Duin, R.P.W.: Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recog. 34(2), 299–314 (2001)
2. Todorovski, L., Džeroski, S.: Combining classifiers with meta decision trees. Mach. Learn. 50(3), 223–249 (2003)
3. Breiman, L.: Bagging predictors. Mach. Learn. 24(2), 123–140 (1996)
4. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: 13th International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann Press, San Francisco (1996)
5. Breiman, L.: Randomizing outputs to increase prediction accuracy. Mach. Learn. 40(3), 229–242 (2000)
6. Ting, K.M., Witten, I.H.: Stacking bagged and dagged models. In: 14th International Conference on Machine Learning, pp. 367–375. Morgan Kaufmann Press, San Francisco (1997)
7. Ting, K.M., Witten, I.H.: Issues in stacked generalization. J. Artif. Intell. Res. 10, 271–289 (1999)
8. Merz, C.J.: Using corresponding analysis to combine classifiers. Mach. Learn. 36(1/2), 33–58 (1999)
9. Seewald, A.K.: How to make stacking better and faster while also taking care of an unknown weakness. In: 19th International Conference on Machine learning, pp. 554–561. Morgan Kaufmann Press, San Francisco (2002)
10. Džeroski, S., Ženko, B.: Is combining classifiers with stacking better than selecting the best ones? Mach. Learn. 54(3), 255–273 (2004)
11. Raudys, S.: Trainable fusion rules: I. Large sample size case. Neural Networks 19(10), 1506–1516 (2006)
12. Raudys, S.: Trainable fusion rules: II. Small sample-size effects. Neural Networks 19(10), 1517–1527 (2006)
13. Paclík, P., Landgrebe, T.C.W., Tax, D.M.J., Duin, R.P.W.: On deriving the second-stage training set for trainable combiners. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 136–146. Springer, Heidelberg (2005)
14. Liu, M., Yuan, B.Z., Chen, J.F., Miao, Z.j.: Does linear combination outperform the k-NN rule? In: 8th International Conference on Signal Processing, vol. 3. IEEE Press, Beijing (2006)
15. Lawson, C.J., Hanson, R.J.: Solving Least Squares Problems. SIAM Publications, Philadephia (1995)
16. Wolpert, D.H.: Stacked generalization. Neural Networks 5(2), 241–259 (1992)
17. UCI machine larning respository, `http://www.ics.uci.edu/~mlearn/MLRespository.html`
18. Lai, C.: Supervised classification and spatial dependency analysis in human cancer using high throughput data. Ph.D Thesis, Delft University of Technology (2008)